Esercizio

- Si vuole realizzare un data warehouse per una azienda che vende mobili all'ingrosso.
- Il data warehouse deve permettere di analizzare i ricavi dell'azienda. Costi e ricavi devono essere analizzati considerando i seguenti parametri:
  - mobili
  - clienti
  - tempo (a livello giorno)
- L'azienda è interessata ad analizzare i mobili rispetto al loro tipo (tavoli, sedia, armadi da camera, armadi da ufficio, ecc) e rispetto alla loro categoria di appartenenza (cucina, soggiorno, camera, bagno, ufficio, etc). I clienti devono potere essere analizzati rispetto alla loro collocazione geografica, considerando almeno città, regione, stati.

Esercizio

- Le vendite e i costi devono essere analizzati considerando le seguenti quantità (queste informazioni possono essere estratte da ogni riga del documento di fatturazione):
  - A = quantità venduta
  - B = prezzo totale (per la quantità considerata)
  - C = sconto

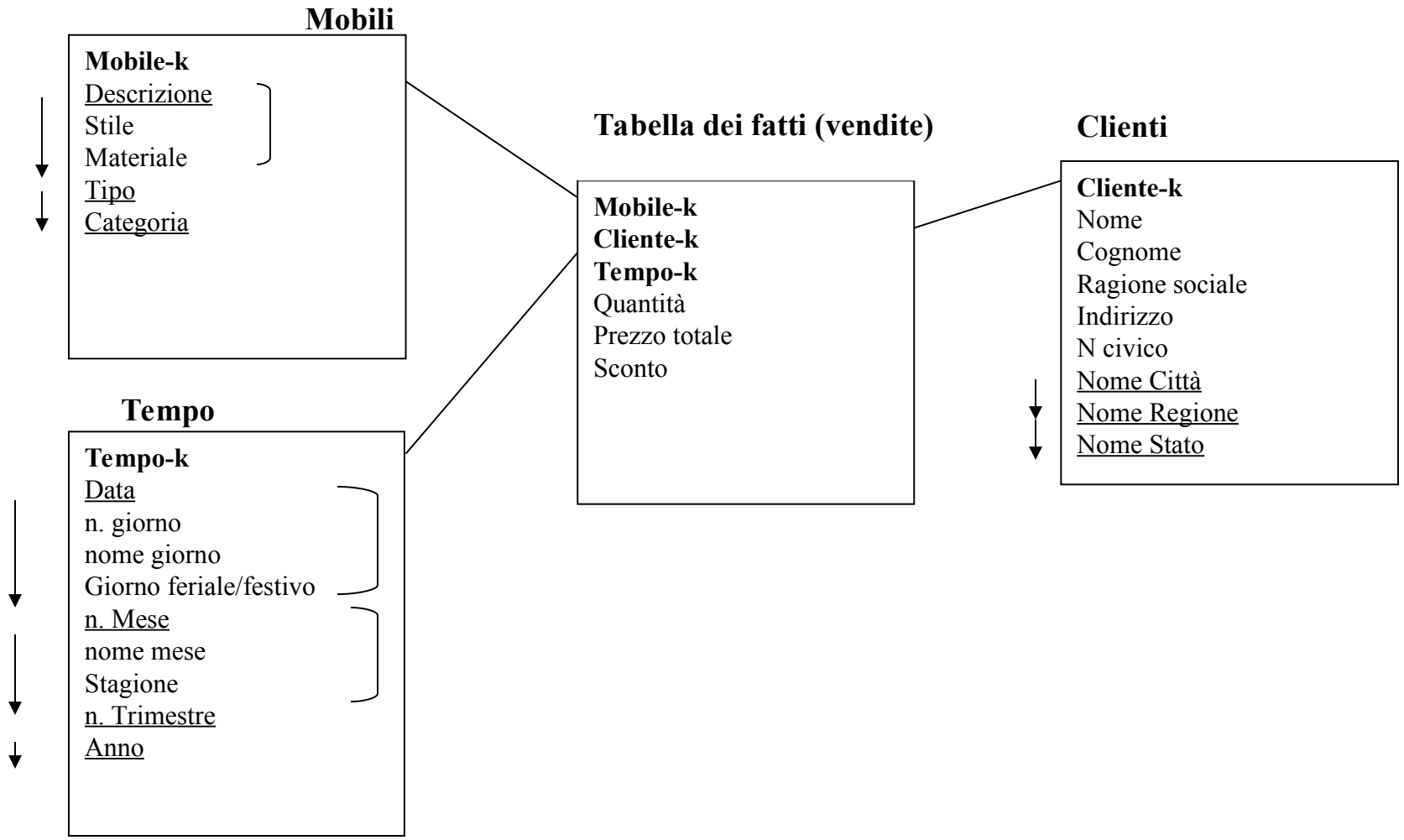
- Si richiede di progettare uno schema a stella per l'attività sopra delineata; a questo proposito:
  - si fissi una granularità per il data warehouse
  - si identifichino le dimensioni, i loro attributi (un insieme minimo), e i fatti
  - Per ciascuna dimensione, mettere inoltre in evidenza le gerarchie che è necessario considerare.

Soluzione

- **Granularità**: a livello di singola linea documento fatturazione (mobile per cliente per tempo gg come risulta da documento di fatturazione).

- Nello schema seguente:
  - gli attributi che identificano i livelli della gerarchia sono sottolineati
  - vengono inoltre inserite delle frecce da ogni livello figlio al livello padre
  - gli attributi associati ad un livello vengono raggruppati insieme all'identificatore del livello

Soluzione

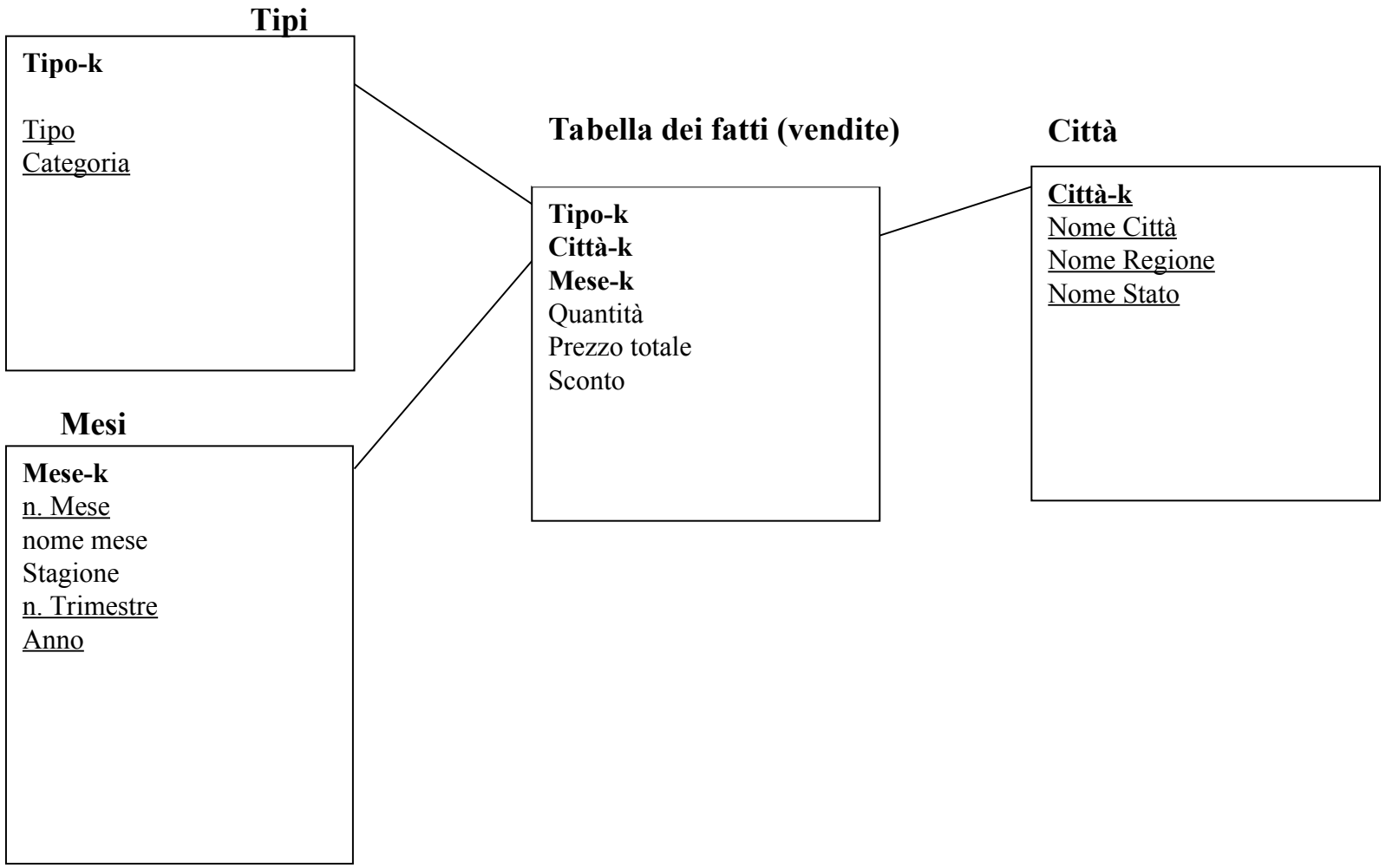

Mobili
Mobile-k
Descrizione
Stile
Materiale
Tipo
Categoria
Tempo
Tempo-k
Data
n. giorno
nome giorno
Giorno feriale/festivo
n. Mese
nome mese
Stagione
n. Trimestre
Anno
Tabella dei fatti (vendite)
Mobile-k
Cliente-k
Tempo-k
Quantità
Prezzo totale
Sconto
Clienti
Cliente-k
Nome
Cognome
Ragione sociale
Indirizzo
N civico
Nome Città
Nome Regione
Nome Stato

Esercizio

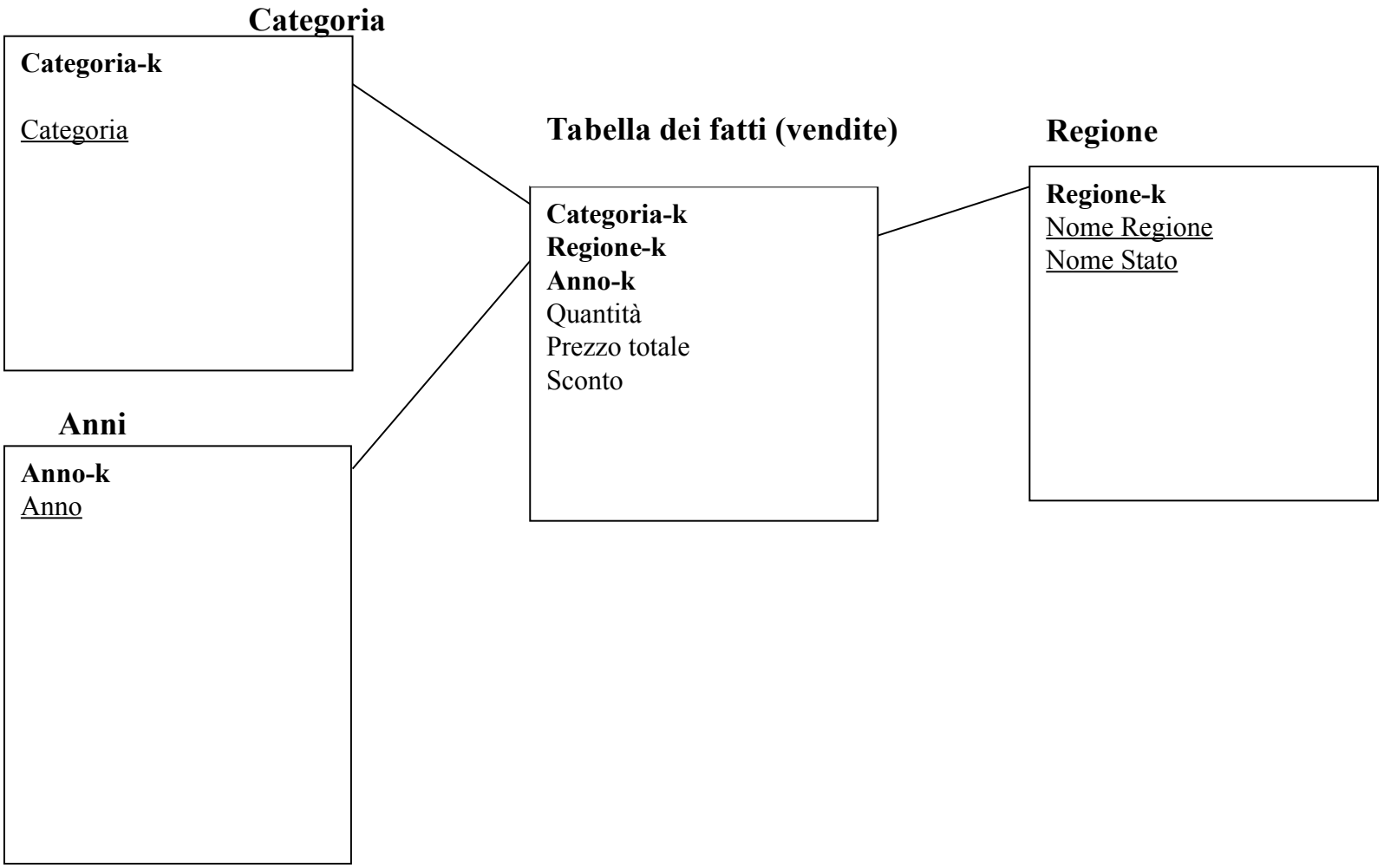
- In riferimento al data warehouse progettato, si supponga di essere a conoscenza che l'azienda abbia interesse ad analizzare i ricavi nel modo seguente:
  1. mensilmente, rispetto a tipo di mobile e città;
  2. annualmente, rispetto alla categoria e rispetto alla regione.

- Si richiede di definire opportunamente gli schemi aggregati corrispondenti alle esigenze di analisi sopra delineate.

Soluzione 1)


Tipi
Tipo-k
Tipo
Categoria
Tabella dei fatti (vendite)
Tipo-k
Città-k
Mese-k
Quantità
Prezzo totale
Sconto
Città
Città-k
Nome Città
Nome Regione
Nome Stato
Mesi
Mese-k
n. Mese
nome mese
Stagione
n. Trimestre
Anno

Soluzione 2)


Categoria
Categoria-k
Categoria
Tabella dei fatti (vendite)
Categoria-k
Regione-k
Anno-k
Quantità
Prezzo totale
Sconto
Regione
Regione-k
Nome Regione
Nome Stato
Anni
Anno-k
Anno

Esercizio

- In riferimento all'ultimo schema aggregato presentato, si supponga che l'azienda venda sia prodotti nuovi che prodotti di seconda mano
- Per analizzare questo tipo di vendite, si supponga di aggiungere i fatti:
  - % mobili di seconda mano (per un certo anno, categoria, regione)
  - % mobili nuovi (per un certo anno, categoria, regione)
- Discutere l'addittività di questi fatti e illustrare come questi fatti possono essere resi addittivi

Soluzione

- I fatti considerati non sono addittivi (le percentuali non si possono sommare)
- Per rendere le informazioni addittivi, ottenendo uno schema semanticamente equivalente si possono aggingere i seguenti fatti:
  - numero mobili nuovi
  - numero mobili di seconda mano
- le percentuali dovranno quindi essere calcolate in fase di interrogazione

Esercizio

- Definire usando SQL di Oracle8i le dimensioni associate alla tabella tempo
- CREATE DIMENSION Tempo_D
  LEVEL giorno_l IS Tempo.Data
  LEVEL mese_l   IS Tempo.n_Mese
  LEVEL trimestre_l IS Tempo.n_Trimestre
  LEVEL anno_l IS Tempo.Anno
  HIERARCHY Tempo_H (
  giorno_lCHILD OF
  mese_l   CHILD OF
  trimestre_l   CHILD OF
  anno_L)
  ATTRIBUTE giorno_l DETERMINES (n_Giorno,
  nome_giorno,giorno_feriale_festivo)
  ATTRIBUTE mese_l DETERMINES (nome_mese, stagione);

Esercizio

- Si definisca usando SQL di Oracle 8i una vista materializzata per supportare la prima aggregazione proposta supponendo che la vista:
  - venga costruita all'atto della definizione
  - venga aggiornata totalmente al commit delle operazioni
  - venga utilizzata dall'aggregate navigator
- Si supponga che non esistano le tabelle delle dimensioni aggregate, quindi definite l'aggregazione a partire dalle tabelle di base

Soluzione

```
CREATE MATERIALIZED VIEW Vendite_mensili_per_tipo_città
    BUILD immediate
    REFRESH complete on commit
    ENABLE query rewrite
AS
SELECT c.Città, t.n_Mese, m.Tipo,
     SUM(Quantita),
     SUM(Prezzo_totale),
     SUM(Sconto)
FROM Vendite v, Clienti c, Tempo t, Mobili m
WHERE v.Cliente_k = c.Cliente_k AND v.Tempo_k = t.Tempo_k AND
    v.Mobile_k = m.Mobile_k
GROUP BY c.Città, t.n_Mese, m.Tipo;
```

Osservazione

- Lo schema proposto durante la progettazione concettuale in generale può essere implementato in due modi:
  - con viste materializzate
    - in questo caso non abbiamo le chiavi artificiali
    - le chiavi diventano gli identificatori dei livelli
  - come schema da alimentare con dati sorgente:
    - in questo caso, le tabelle delle dimensioni e dei fatti aggregate non sono viste materializzate ma tabelle come le altre, nelle quali verranno caricati dati aggregati

Esercizio

- In riferimento allo schema di base proposto in precedenza, scrivere il codice SQL corrispondente alle seguenti interrogazioni OLAP:
- ROLLUP: determinare le quantità, i ricavi e gli sconti totali rispetto alle città, il tipo del mobile e al mese. Si vuole inoltre realizzare un'operazione di rollup che permetta di analizzare i fatti anche rispetto al mese e al tipo e solo al mese.

```
SELECT c.citta, m.tipo, t.mese,
    SUM(v.quantita), SUM(v.Prezzo_totale), SUM(v.Sconto)
FROM Vendite v, Tempo t, Clienti c, Mobili  m
WHERE v.Cliente_k = c.Cliente_k AND
            v.Tempo_k = t.Tempo_k AND
            v.Mobile_k = m.Mobile_k
GROUP BY ROLLUP(t.mese,m.tipo,c.citta);
```

Esercizio

- ROLLUP: determinare le quantità, i ricavi e gli sconti medi rispetto alle regioni, al materiale e all'anno di vendita. Si vuole inoltre realizzare un'operazione di rollup nel seguente ordine: anno, regione, materiale.

```
SELECT c.regione, m.materiale, t.anno,
    SUM(v.quantita), SUM(v.Prezzo_totale), SUM(v.Sconto)
FROM Vendite v, Tempo t, Clienti c, Mobili  m
WHERE v.Cliente_k = c.Cliente_k AND
            v.Tempo_k = t.Tempo_k AND
            v.Mobile_k = m.Mobile_k
GROUP BY ROLLUP(t.anno,c.regione,m.materiale);
```

Esercizio

- CUBE: determinare determinare le quantità, i ricavi e gli sconti totali rispetto alle città, il tipo del mobile e al mese, in ogni possibile combinazione.

```
SELECT c.citta, m.tipo, t.mese,
     SUM(v.quantita), SUM(v.Prezzo_totale), SUM(v.Sconto)
FROM Vendite v, Tempo t, Clienti c, Mobili  m
WHERE v.Cliente_k = c.Cliente_k AND
              v.Tempo_k = t.Tempo_k AND
              v.Mobile_k = m.Mobile_k
GROUP BY CUBE(t.mese,m.tipo,c.citta);
```

Esercizio

- CUBE: determinare le quantità, i ricavi e gli sconti medi rispetto all'anno di vendita, alla regione e al materiale, in ogni possibile combinazione.

```
SELECT c.regione, m.materiale, t.anno,
            SUM(v.quantita), SUM(v.Prezzo_totale), SUM(v.Sconto)
FROM Vendite v, Tempo t, Clienti c, Mobili  m
WHERE v.Cliente_k = c.Cliente_k AND
        v.Tempo_k = t.Tempo_k AND
        v.Mobile_k = m.Mobile_k
GROUP BY CUBE(t.anno,c.regione,m.materiale);
```

Esercizio

- TOP-N: determinare i 5 mobili che sono stati più venduti (come quantità) nel mese di maggio

```
SELECT m.descrizione,  sum_quantità
FROM
    (SELECT m.descrizione, SUM(quantità) as sum_quantità,
                        RANK() OVER (ORDER BY SUM(quantità)  DESC)  AS
    rank
     FROM vendite v, mobili m, tempo t
     WHERE  v.Mobile_k = m.Mobile_k AND
                        v.Tempo_k = t.Tempo_k  AND
                t.nome_mese = “Maggio”)
WHERE RANK <= 5;
```